

姚攀的博客 1.01^365=31.78

Engineers are versatile minds who create links between science, technology, and society.

目录视图

摘要视图

RSS 订阅

«从Lucene到Elasticsearch：全文检索实战»

希望本书能对您的工作和学习带来帮助，感谢支持！



当当 京东 亚马逊 天猫

Lucene、ES、ELK开发交流群:370734940

 加入QQ群

个人资料



yyyseb

关注 发私信

博客

掘金

访问： 844557次

积分： 8642

等级： 

排名： 第2540名

图灵赠书——程序员11月书单 【思考】Python这么厉害的原因竟然是！ 感恩节赠书：《深度学习》等异步社区优秀图书和作译者评选启动！ 每周荐书：京东架构、Linux内核、Python全栈

(spring全家桶十)Spring Statemachine有限状态机与地址分析

标签： 有限状态机 state machine spring

2017-09-23 15:47 897人阅读 评论(0)

分类： J2EE (34)

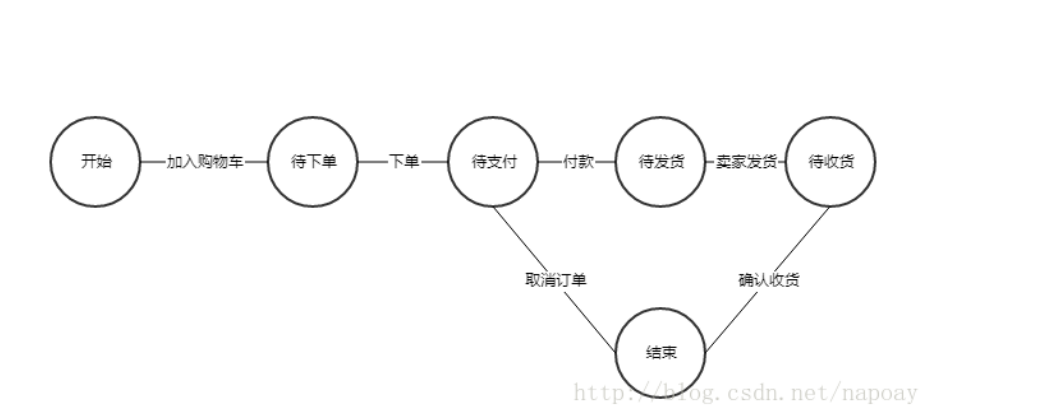
版权声明：本文为博主原创文章,未经博主允许禁止转载(<http://blog.csdn.net/napoay>)

目录(?)

一、有限状态机

有限状态机是一个特殊的有向图，包含节点和连接这些节点的弧。每个有限状态机都有开始、结束和若干个中间状态，每个弧上带有从一个状态进入下一个状态的条件。

以一个简化的购物流程为例，开始和结束之间有待下单、待支付、待发货、待收货四个状态，从一个状态转向另外一个状态中间需要发送事件。



有限状态机可以用于中文地址分析，识别地址的有限状态机如下。

原创： 202篇      转载： 2篇

译文： 6篇      评论： 451条

我的课程

Elasticsearch 5.4新闻搜索项目实战

StackOverFlow

<http://stackoverflow.com/users/6526424>  
统计

阅读排行

[搜索]ElasticSearch Java Api(... (61756)

Elasticsearch索引mapping的写... (36806)

Elasticsearch 5.X下JAVA API使... (31047)

搭建Elasticsearch 5.4分布式集群 (29631)

Elasticsearch 5.1.1 head插件安... (25822)

Elasticsearch java api(五) Bulk... (21822)

ElasticSearch Java Api(四)-删... (19887)

mac命令行启动tomcat (18615)

Elasticsearch 5 Ik+pinyin分词... (17900)

Elasticsearch shield权限管理详解 (17369)

博客专栏

Spring全家桶

文章： 10篇  
阅读： 10622

MapReduce编程

文章： 7篇  
阅读： 24325

Lucene&Elasticsearch&ELK 专栏

文章： 42篇  
阅读： 461286

数据库

文章： 9篇  
阅读： 18151

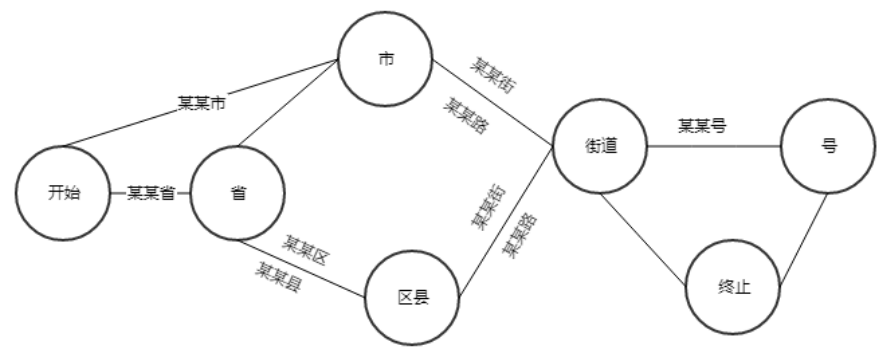
J2EE

文章： 24篇  
阅读： 92799

ruby on rails

文章： 18篇  
阅读： 37381

数据结构与算法



识别地址的有限状态机 <http://blog.csdn.net/>

给出一个地址，如果当前状态是“省”，后面一个词组是二级市，就进入状态“市”，后面一个词组是某某区或者某某县，就进入“区县”状态。

如果一个地址能从状态机的开始状态经过若干中间状态到达终止状态，地址有效，否则无效。

例如：北京市海淀区清华东路17号，有效。河北省大连市友好路，无效。

## 二、Spring Statemachine

Spring Statemachine是spring的有限状态机开源框架，诞生不久。

Spring Statemachine框架的目标：

- 1.简单易用
- 2.采用层次化状态机结构简化复杂状态配置
- 3.State machine regions提供复杂的状态机配置
- 4.使用triggers, transitions, guards and actions.
- 5.类型安全的适配器配置
- 6.用于Spring应用程序上下文之外的简单实例化的生成器模式
- 7.基于ZooKeeper实现的分布式状态机
- 8.有事件监听器
- 9.在持久层存储状态配置
- 10.Spring IOC集成，bean可以和状态机交互

## 三、实例

### 3.1 创建spring boot工程

访问：<http://start.spring.io/>，新建一个spring boot工程：



文章：13篇

阅读：17571

文章分类

Elasticsearch (43)

Lucene (15)

hadoop (16)

J2EE (35)

数据结构 (16)

前端 (16)

ruby (18)

机器学习 (4)

python (5)

linux (11)

latex (2)

数据库 (12)

spring (9)

tools (1)

hibernate (1)

mybatis (1)

生活 (1)

文章存档

2017年12月 (9)

2017年11月 (1)

2017年10月 (3)

2017年09月 (2)

2017年08月 (1)

展开

最新评论

MapReduce编程(六) 从HDFS导入数据到El... qq\_38094271 : 用最新版本的那个 成功了。 注意json 文件最后一行要换行

《从Lucene到Elasticsearch:全文检索实战... 牙小木 : 书已买, 正在看

Lucene扩展停用词字典与自定义词库 yyyseb : @cuipeng6561:这个文件在ik的目录下。

Lucene扩展停用词字典与自定义词库 cuipeng6561 : IKAnalyzer.cfg.xml 这个文件是要自己写么?

新人千万不要在 Windows 上使用 Ruby on ... 韩大喵 : 对于新手的我来说的, 相当中肯!

《从Lucene到Elasticsearch:全文检索实战... 艾鹤 : 6666, 恭喜发财。

Elasticsearch 5.X下JAVA API使用指南 qq\_41296216 : @napoay:maven坐标是 &lt;de pendency>...

Elasticsearch java api(五) Bulk批量索引 yyyseb : @wangmaohong0717:可以不指定, 会自动生成。

Elasticsearch 5.X下JAVA API使用指南 yyyseb : @qq\_41296216:贴一下maven坐标吧, 或者, 博客上有QQ群, 加群讨论

start.spring.io

SPRING INITIALIZR bootstrap your application now

Generate a

Maven Project

with

Java

and Spring Boot

1.5.7

Project Metadata

Artifact coordinates

Group

com.statemachine

Artifact

statemachinedemo

Generate Project alt + ⌘

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web, Security, JPA, Actuator, Devtools...

Selected Dependencies

Don't know what to look for? Want more options? Switch to the full version.

start.spring.io is powered by Spring Initializr and Pivotal Web Services http://blog.csdn

pom.xml:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.c
4     <modelVersion>4.0.0</modelVersion>
5
6     <groupId>com.statemachine</groupId>
7     <artifactId>statemachinedemo</artifactId>
8     <version>0.0.1-SNAPSHOT</version>
9     <packaging>jar</packaging>
10
11     <name>statemachinedemo</name>
12     <description>Demo project for Spring Boot</description>
13
14     <parent>
15         <groupId>org.springframework.boot</groupId>
16         <artifactId>spring-boot-starter-parent</artifactId>
17         <version>1.5.7.RELEASE</version>
18         <relativePath/> <!-- lookup parent from repository -->
19     </parent>
20
21     <properties>
22         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
23         <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncodi
24         <java.version>1.8</java.version>
25     </properties>
26
27     <dependencies>
28
29         <dependency>
30             <groupId>org.springframework.boot</groupId>
31             <artifactId>spring-boot-starter</artifactId>
32         </dependency>
33
34         <dependency>
35             <groupId>org.springframework.boot</groupId>
36             <artifactId>spring-boot-starter-test</artifactId>
37             <scope>test</scope>
```

Elasticsearch java api(五) Bulk批量索引  
hello-friend : 批量索引数据的时候, 必须指定\_id吗

```
38         </dependency>
39
40         <dependency>
41             <groupId>org.springframework.statemachine</groupId>
42             <artifactId>spring-statemachine-core</artifactId>
43             <version>1.2.6.RELEASE</version>
44         </dependency>
45
46         <dependency>
47             <groupId>log4j</groupId>
48             <artifactId>log4j</artifactId>
49             <version>1.2.17</version>
50         </dependency>
51
52     </dependencies>
53
54     <build>
55         <plugins>
56             <plugin>
57                 <groupId>org.springframework.boot</groupId>
58                 <artifactId>spring-boot-maven-plugin</artifactId>
59             </plugin>
60         </plugins>
61     </build>
62
63
64 </project>
```

创建状态:

```
1 public enum States {
2     START,        //开始
3     PROVINCE,     //省
4     CITY,         //市
5     DISTRIC,      //区县
6     STREET,       //街道
7     STREET_NUM,   //街道号
8 }
```

定义事件:

```
1 public enum Events {
2     HAS_PROVINCE, //地址中有一级省份
3     HAS_CITY,     //地址中有二级市
4     HAS_DISTRIC,  //地址中有三级县或者区
5     HAS_STREET,   //地址中有四级的街道
6     HAS_STREET_NUM //地址中有五级的街道号
7 }
```

设置状态之间的转换关系:

```
1 import com.sun.org.apache.regexp.internal.RE;
2 import org.apache.log4j.Logger;
```

```

3  import org.springframework.context.annotation.Configuration;
4  import org.springframework.statemachine.config.EnableStateMachine;
5  import org.springframework.statemachine.config.EnumStateMachineConfigurerAdapter;
6  import org.springframework.statemachine.config.builders.StateMachineConfigurati
7  import org.springframework.statemachine.config.builders.StateMachineStateConfig
8  import org.springframework.statemachine.config.builders.StateMachineTransitionC
9  import org.springframework.statemachine.listener.StateMachineListener;
10 import org.springframework.statemachine.listener.StateMachineListenerAdapter;
11 import org.springframework.statemachine.transition.Transition;
12
13 import java.util.EnumSet;
14
15
16 @Configuration
17 @EnableStateMachine
18 public class Config extends EnumStateMachineConfigurerAdapter<States, Events> {
19     public static final Logger logger = Logger.getLogger(Config.class);
20
21     @Override
22     public void configure(StateMachineStateConfigurer<States, Events> states) throws Exception {
23         states
24             .withStates()
25             .initial(States.START)
26             .states(EnumSet.allOf(States.class));
27     }
28
29     @Override
30     public void configure(StateMachineTransitionConfigurer<States, Events> transitions) throws Exception {
31         transitions
32             .withExternal().source(States.START).target(States.PROVINCE).event("开始")
33             .and()
34             .withExternal().source(States.START).target(States.CITY).event("省")
35             .and()
36             .withExternal().source(States.PROVINCE).target(States.CITY).event("市")
37             .and()
38             .withExternal().source(States.PROVINCE).target(States.DISTRICT).event("区")
39             .and()
40             .withExternal().source(States.CITY).target(States.STREET).event("街道")
41             .and()
42             .withExternal().source(States.STREET).target(States.STREET_NUMBER).event("门牌");
43     }
44
45     public StateMachineListener<States, Events> listener() {
46         return new StateMachineListenerAdapter<States, Events>() {
47             @Override
48             public void transition(Transition<States, Events> transition) {
49                 if (transition.getTarget().getId() == States.START) {
50                     logger.info("开始:");
51                     return;
52                 }
53
54                 if (transition.getSource().getId() == States.START &&
55                     transition.getTarget().getId() == States.PROVINCE) {
56                     logger.info("省份:");
57                     return;
58                 }

```

```

59
60
61         if (transition.getSource().getId() == States.START &&
62             transition.getTarget().getId() == States.CITY) {
63             logger.info("市:");
64             return;
65         }
66
67         if (transition.getSource().getId() == States.PROVINCE &&
68             transition.getTarget().getId() == States.CITY) {
69             logger.info("市:");
70             return;
71         }
72
73         if (transition.getSource().getId() == States.PRC
74             transition.getTarget().getId() == States.DISTRICT) {
75             logger.info("县:");
76             return;
77         }
78
79         if (transition.getSource().getId() == States.CITY &&
80             transition.getTarget().getId() == States.STREET) {
81             logger.info("街道:");
82             return;
83         }
84
85         if (transition.getSource().getId() == States.DISTRIC &&
86             transition.getTarget().getId() == States.STREET) {
87             logger.info("街道:");
88             return;
89         }
90
91         if (transition.getSource().getId() == States.STREET &&
92             transition.getTarget().getId() == States.STREET_NUM) {
93             logger.info("街道号:");
94             return;
95         }
96     }
97     };
98 }
99
100 @Override
101 public void configure(StateMachineConfigurationConfigurer<States, Events> c
102     config.withConfiguration().listener(listener());
103
104 }
105 }

```

运行:

```

1 import org.springframework.beans.factory.annotation.Autowired;
2 import org.springframework.boot.CommandLineRunner;
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.statemachine.StateMachine;

```

```

6  @SpringBootApplication
7  public class Application implements CommandLineRunner {
8
9      public static void main(String[] args) {
10         SpringApplication.run(Application.class, args);
11     }
12
13     @Autowired
14     private StateMachine<States, Events> stateMachine;
15
16     @Override
17     public void run(String... args) throws Exception {
18         stateMachine.start();
19         stateMachine.sendEvent(Events.HAS_PROVINCE);
20         stateMachine.sendEvent(Events.HAS_CITY);
21         stateMachine.sendEvent(Events.HAS_STREET);
22         stateMachine.sendEvent(Events.HAS_STREET_NUM);
23     }
24 }

```

结果:

```

1
2      .  _____  -                _ _ _ _
3  /\ /  /  ___ ' _ _ _ _ _ ( ) _ _ _ _ _ \ \ \ \
4  ( ( ) \___ | ' _ | ' _ | | ' _ \ / _ \ | \ \ \ \
5  \ \ /  ___ ) | | _ | | | | | | ( _ | | ) ) ) )
6  '   | ___ | . _ | | | _ | | _ \ , / / / /
7  =====|_|=====|___/=/_/_/_/
8  :: Spring Boot ::                (v1.5.6.RELEASE)
9
10 2017-09-23 17:55:27.787 INFO 9344 --- [           main] xtipc.statemachine.adc
11 2017-09-23 17:55:27.790 INFO 9344 --- [           main] xtipc.statemachine.adc
12 2017-09-23 17:55:27.864 INFO 9344 --- [           main] s.c.a.AnnotationConfi
13 2017-09-23 17:55:28.565 INFO 9344 --- [           main] trationDelegate$BeanPc
14 2017-09-23 17:55:29.075 INFO 9344 --- [           main] o.s.j.e.a.AnnotationMI
15 2017-09-23 17:55:29.079 INFO 9344 --- [           main] o.s.c.support.DefaultI
16 2017-09-23 17:55:29.089 INFO 9344 --- [           main] xtipc.statemachine.adc
17 2017-09-23 17:55:29.092 INFO 9344 --- [           main] o.s.s.support.Lifecyc
18 2017-09-23 17:55:29.092 INFO 9344 --- [           main] o.s.s.support.Lifecyc
19 2017-09-23 17:55:29.097 INFO 9344 --- [           main] xtipc.statemachine.adc
20 2017-09-23 17:55:29.098 INFO 9344 --- [           main] xtipc.statemachine.adc
21 2017-09-23 17:55:29.098 INFO 9344 --- [           main] xtipc.statemachine.adc
22 2017-09-23 17:55:29.098 INFO 9344 --- [           main] xtipc.statemachine.adc
23 2017-09-23 17:55:29.099 INFO 9344 --- [           main] xtipc.statemachine.adc
24 2017-09-23 17:55:29.100 INFO 9344 --- [ Thread-2] s.c.a.AnnotationConfi
25 2017-09-23 17:55:29.101 INFO 9344 --- [ Thread-2] o.s.c.support.DefaultI
26 2017-09-23 17:55:29.102 INFO 9344 --- [ Thread-2] o.s.s.support.Lifecyc
27 2017-09-23 17:55:29.102 INFO 9344 --- [ Thread-2] o.s.s.support.Lifecyc
28 2017-09-23 17:55:29.103 INFO 9344 --- [ Thread-2] o.s.j.e.a.AnnotationMI
29 2017-09-23 17:55:29.112 INFO 9344 --- [ Thread-2] o.s.s.support.Lifecyc

```

顶

1

踩

0

- [上一篇](#) [Elasticsearch 集群优化总结](#)
- [下一篇](#) [Spark机器学习环境搭建](#)

相关文章推荐

- [基于Spring-statemachine的有限状态机\(FSM\)的介绍...](#)
- [MySQL在微信支付下的高可用运营--莫晓东](#)
- [使用Spring StateMachine框架实现状态机](#)
- [容器技术在58同城的实践--姚远](#)
- [使用Spring StateMachine框架实现状态机](#)
- [SDCC 2017之容器技术实战线上峰会](#)
- [使用Spring StateMachine框架实现状态机](#)
- [SDCC 2017之数据库技术实战线上峰会](#)
- [数学之美系列十：有限状态机和地...](#)
- [腾讯云容器服务架构实现介绍--董明](#)
- [Android蓝牙源码分析——StateMachine状态机](#)
- [微博热点事件背后的数据库运维心得 一...](#)
- [quick-cocos2d-x游戏开发——StateMa](#)
- [StateMachine状态机](#)
- [Mina状态机介绍\(Introduction to mina-statemachine\)](#)
- [quick-cocos2d-x游戏开发——StateMachine状态机](#)

查看评论

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#)   [杂志客服](#)   [微博客服](#)   [webmaster@csdn.net](#)   400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved

